MARKED UP VERSION INDICATING CHANGES MADE

SUBSTITUTE SPECIFICATION

JAVA™ BYTECODE INSTRUCTION FOR RETRIEVING STRING REPRESENTATIONS OF JAVA™ OBJECTS

JAVA™ ~~JAVA~~ BYTECODE INSTRUCTION FOR RETRIEVING STRING REPRESENTATIONS OF JAVA™ ~~JAVA~~ OBJECTS

BACKGROUND OF THE INVENTION

[0001] The present invention relates generally to JAVA™ ~~Java~~ programming environments, and more particularly, to techniques suitable for retrieving string representations of JAVA™ ~~Java~~ objects.

[0002] One of the goals of high level languages is to provide a portable programming environment such that the computer programs may easily be ported to another computer platform. High level languages such as "C" provide a level of abstraction from the underlying computer architecture and their success is well evidenced from the fact that most computer applications are now written in a high level language.

[0003] Portability has been taken to new heights with the advent of the World Wide Web ("the Web") which is an interface protocol for the Internet which allows communication between diverse computer platforms through a graphical interface. Computers communicating over the Web are able to download and execute small applications called applets. Given that applets may be executed on a diverse assortment of computer platforms, the applets are typically executed by a JAVA™ ~~Java~~ virtual machine.

[0004] Recently, the JAVA™ ~~Java~~ programming environment has become quite popular. The JAVA™ ~~Java~~ programming language is a language that is designed to be portable enough to be executed on a wide range of computers ranging from small devices (e.g., pagers, cell phones and smart cards) up to supercomputers. Computer programs written in the JAVA™ ~~Java~~ programming language (and other languages) may be compiled into JAVA™ ~~Java~~ Bytecode instructions that are suitable for execution by a JAVA™ ~~Java~~ virtual machine implementation. The JAVA™ ~~Java~~ virtual machine is commonly implemented in software by means of an interpreter for the JAVA™ ~~Java~~ virtual machine instruction set but, in general, may be software, hardware, or both. A particular

JAVA™~~Java~~ virtual machine implementation and corresponding support libraries together constitute a JAVA™~~Java~~ runtime environment.

[0005] Computer programs in the JAVA™~~Java~~ programming language are arranged in one or more classes or interfaces (referred to herein jointly as classes or class files). Such programs are generally platform, i.e., hardware and operating system, independent. As such, these computer programs may be executed without modification on any computer that is able to run an implementation of the JAVA™~~Java~~ runtime environment.

[0006] Object-oriented classes written in the JAVA™~~Java~~ programming language are compiled to a particular binary format called the "class file format." The class file includes various components associated with a single class. These components can be, for example, methods and/or interfaces associated with the class. In addition, the class file format can include a significant amount of ancillary information that is associated with the class. The class file format (as well as the general operation of the JAVA™~~Java~~ virtual machine) is described in some detail in The JAVA™~~Java~~ Virtual Machine Specification, Second Edition, by Tim Lindholm and Frank Yellin, which is hereby incorporated herein by reference.

[0007] Fig. 1A shows a progression of a simple piece of a JAVA™~~Java~~ source code 101 through execution by an interpreter, the JAVA™~~Java~~ virtual machine. The JAVA™~~Java~~ source code 101 includes the classic Hello World program written in JAVA™~~Java~~. The source code is then input into a Bytecode compiler 103 that compiles the source code into Bytecodes. The Bytecodes are virtual machine instructions as they will be executed by a software emulated computer. Typically, virtual machine instructions are generic (i.e., not designed for any specific microprocessor or computer architecture) but this is not required. The Bytecode compiler outputs a JAVA™~~Java~~ class file 105 that includes the Bytecodes for the JAVA™~~Java~~ program. The JAVA™~~Java~~ class file is input into a JAVA™~~Java~~ virtual machine 107. The JAVA™~~Java~~ virtual machine is an interpreter that decodes and executes the Bytecodes in the JAVA™~~Java~~ class file. The JAVA™~~Java~~ virtual machine is an interpreter, but is commonly referred to as a virtual machine as it emulates a microprocessor

or computer architecture in software (e.g., the microprocessor or computer architecture may not exist in hardware).

[0008] Fig. 1B illustrates a simplified class file 100. As shown in Fig. 1B, the class file 100 includes a constant pool 102 portion, interfaces portion 104, fields portion 106, methods portion 108, and attributes portion 110. The methods portion 108 can include or have references to several JAVA™~~Java~~ methods associated with the JAVA™~~Java~~ class which is represented in the class file 100.

[0009] As is known in the art, often there is a need to represent a JAVA™~~Java~~ object as a string of characters. For example, in order to print an integer object, (or an integer field of an object) there is a need to represent the integer as a string of characters. Conventionally, a JAVA™~~Java~~ method, "JAVA™~~Java~~.lang.object.to_string()" is invoked by the virtual machine to represent objects (or fields associated with objects) as a string of characters. One problem with this approach is that there is an overhead associated with the invocation of a JAVA™~~Java~~ method. In other words, invocation of a JAVA™~~Java~~ method requires several operations to be performed. These operations include: locating the appropriate method to be invoked, creating a frame to be placed on the execution stack and restoring the previous frame on the stack.

[0010] Moreover, the cost associated with representing JAVA™~~Java~~ objects as strings is quite high because, during execution of a typical JAVA™~~Java~~ program, the to_string JAVA™~~Java~~ method has to be invoked time and time again. In other words, the operations needed to invoke a method have to be performed several times during the execution of a JAVA™~~Java~~ program. This, of course, can result in a grossly inefficient use of system resources. In some circumstances, particularly in systems with limited computing power and/or memory, this inefficient use of resources is a serious disadvantage.

[0011] In view of the foregoing, improved techniques for retrieving string representations of JAVA™~~Java~~ objects are needed.

SUMMARY OF THE INVENTION

[0012] Broadly speaking, the invention relates to improved techniques for representing JAVA™~~Java~~ objects as strings. In accordance with one aspect of the invention, an inventive JAVA™~~Java~~ Bytecode instruction suitable for execution by a JAVA™~~Java~~ virtual machine is disclosed. As such, the inventive JAVA™~~Java~~ Bytecode instruction can be executed by a JAVA™~~Java~~ virtual machine to represent JAVA™~~Java~~ objects as strings. Moreover, the JAVA™~~Java~~ objects can be represented as strings without invoking the JAVA™~~Java~~ "to_string" method which is conventionally used. This means that the costly overhead associated with repeatedly invoking JAVA™~~Java~~ method "to_string" is avoided. In other words, operations that are conventionally performed each time the JAVA™~~Java~~ "to_string" method is invoked need not be performed. As a result, the performance of virtual machines, especially those operating with limited resources (e.g., embedded systems) can be improved.

[0013] The invention can be implemented in numerous ways, including as a method, an apparatus, a computer readable medium, and a database system. Several embodiments of the invention are discussed below.

[0014] As a JAVA™~~Java~~ Bytecode instruction suitable for execution by a JAVA™~~Java~~ virtual machine in the JAVA™~~Java~~ computing environment, one embodiment of the invention operates to retrieve a string representation associated with the JAVA™~~Java~~ object, thereby allowing the string representation to be determined without invoking a JAVA™~~Java~~ method.

[0015] As a JAVA™~~Java~~ virtual machine operating in a JAVA™~~Java~~ computing environment, one embodiment of the invention includes a JAVA™~~Java~~ virtual machine capable of determining a string representation associated with a JAVA™~~Java~~ object. The virtual machine determines the string representation of the JAVA™~~Java~~ object without invoking a JAVA™~~Java~~ "to_string" method.

[0016] As a method for retrieving a string representation for a JAVA™~~Java~~ object, one embodiment of the invention includes the acts of:

receives an inventive JAVA™~~Java~~ Bytecode instruction in a stream of JAVA™~~Java~~ Bytecodes suitable for execution by a virtual machine operating in the JAVA™~~Java~~ computing environment. The JAVA™~~Java~~ Bytecode instruction operates to determine the string representation associated with the JAVA™~~Java~~ object; thereby allowing the string representation to be determined without invoking a JAVA™~~Java~~ method.

[0017] As a computer readable media including computer program code for retrieving a string representation for a JAVA™~~Java~~ object, one embodiment of the invention includes computer program code for receiving an inventive JAVA™~~Java~~ Bytecode instruction in a stream of JAVA™~~Java~~ Bytecodes suitable for execution by a virtual machine operating in the JAVA™~~Java~~ computing environment. The inventive JAVA™~~Java~~ Bytecode instruction operates to determine the string representation associated with the JAVA™~~Java~~ object, thereby allowing the string representation to be determined without invoking a JAVA™~~Java~~ method.

[0018] These and other aspects and advantages of the present invention will become more apparent when the detailed description below is read in conjunction with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

[0019] The present invention will be readily understood by the following detailed description in conjunction with the accompanying drawings, wherein like reference numerals designate like structural elements, and in which:

Fig. 1A shows a progression of a simple piece of a **JAVA™**~~Java~~ source code through execution by an interpreter, the **JAVA™**~~Java~~ virtual machine.

Fig. 1B illustrates a simplified class file.

Figs. 2A-2B illustrate a **JAVA™**~~Java~~ computing environment in accordance with one embodiment of the invention.

Fig. 3 illustrates a method for representing **JAVA™**~~Java~~ objects as string in accordance with one embodiment of the invention.

DETAILED DESCRIPTION OF THE INVENTION

[0020] The present invention pertains to improved techniques for representing JAVA™~~Java~~ objects as strings. In accordance with one aspect of the invention, an inventive JAVA™~~Java~~ Bytecode instruction suitable for execution by a JAVA™~~Java~~ virtual machine is disclosed. As such, the inventive JAVA™~~Java~~ Bytecode instruction can be executed by a JAVA™~~Java~~ virtual machine to represent JAVA™~~Java~~ objects as strings. Moreover, the JAVA™~~Java~~ objects can be represented as strings without invoking the JAVA™~~Java~~ "to_string" method which is conventionally used. This means that the costly overhead associated with repeatedly invoking JAVA™~~Java~~ method "to_string" is avoided. In other words, operations that are conventionally performed each time the JAVA™~~Java~~ "to_string" method is invoked need not be performed. As a result, the performance of virtual machines, especially those operating with limited resources (e.g., embedded systems) can be improved.

[0021] Embodiments of the invention are discussed below with reference to Figs. 2A-3. However, those skilled in the art will readily appreciate that the detailed description given herein with respect to these figures is for explanatory purposes only as the invention extends beyond these limited embodiments.

[0022] Figs. 2A-B illustrate a JAVA™~~Java~~ computing environment 200 including a virtual machine 202 in accordance with one embodiment of the invention. The virtual machine 202 can read a stream of JAVA™~~Java~~ Bytecodes 204 as input. The stream of JAVA™~~Java~~ Bytecodes 204 includes a JAVA™~~Java~~ "Aload" Bytecode instruction 206 and an inventive JAVA™~~Java~~ "to_string" Bytecode 208. The "Aload" Bytecode instruction 206 can be implemented as a JAVA™~~Java~~ Bytecode instruction which operates to push a reference A to a JAVA™~~Java~~ object 210 on an execution stack 212.

[0023] The inventive JAVA™~~Java~~ "to_string" Bytecode instruction 208 is a JAVA™~~Java~~ Bytecode instruction that has been specifically designated

for representing JAVA™~~Java~~ objects as strings. As will be appreciated, the inventive JAVA™~~Java~~ "to_string" Bytecode instruction can be implemented as a new instruction that is added to the conventional JAVA™~~Java~~ Bytecode instruction set. This is possible because the conventional JAVA™~~Java~~ Bytecode instruction set does not typically use all the 256 possible values that can be coded by one byte (8 bits). As such, the inventive JAVA™~~Java~~ "to_string" Bytecode instruction set can be assigned a unique unassigned value which can be represented by 8 bits.

[0024] Referring now to Fig. 2B, the virtual machine 202 operates to receive the inventive JAVA™~~Java~~ Bytecode instruction 208. When the inventive JAVA™~~Java~~ "to_string" Bytecode instruction 208 is executed, the string representation of the JAVA™~~Java~~ object referenced by reference A (shown in Fig. 2A) is determined. Accordingly, the inventive JAVA™~~Java~~ Bytecode instruction "to_string" 208 can be used to represent JAVA™~~Java~~ objects as strings. Moreover, JAVA™~~Java~~ objects can be represented as strings without invoking a JAVA™~~Java~~ method. This means that costly overhead associated with invoking JAVA™~~Java~~ methods can be avoided. As a result, the performance of virtual machines, especially those operating with limited resources, can be improved.

[0025] Fig. 3 illustrates a method 300 for representing JAVA™~~Java~~ objects as string in accordance with one embodiment of the invention. The method 300 can be implemented by a virtual machine operating in a JAVA™~~Java~~ computing environment. Initially, at operation 302, a reference to a JAVA™~~Java~~ object is pushed on the execution stack. Next, at operation 304, an inventive JAVA™~~Java~~ Bytecode operation is executed. The inventive JAVA™~~Java~~ Bytecode operation is designated to represent the object as a string. Accordingly, at operation 306, the string representation of the JAVA™~~Java~~ object is determined using the reference to the JAVA™~~Java~~ object. Thereafter, at operation 308, the reference is popped from the stack. Finally, at operation 310, the string representation of the JAVA™~~Java~~ object is pushed on the top of the execution stack.

[0026] The many features and advantages of the present invention are apparent from the written description, and thus, it is intended by the

appended claims to cover all such features and advantages of the invention. Further, since numerous modifications and changes will readily occur to those skilled in the art, it is not desired to limit the invention to the exact construction and operation as illustrated and described. Hence, all suitable modifications and equivalents may be resorted to as falling within the scope of the invention.

What is claimed is:

JAVA™~~JAVA~~ BYTECODE INSTRUCTION FOR RETRIEVING STRING REPRESENTATIONS OF JAVA™~~JAVA~~ OBJECTS

ABSTRACT OF THE DISCLOSURE

Improved techniques for representing JAVA™~~Java~~ objects as strings are disclosed. An inventive JAVA™~~Java~~ Bytecode instruction suitable for execution by a JAVA™~~Java~~ virtual machine is disclosed. The inventive JAVA™~~Java~~ Bytecode instruction can be executed by a JAVA™~~Java~~ virtual machine to represent JAVA™~~Java~~ objects as strings. Moreover, JAVA™~~Java~~ objects can be represented as strings without invoking the JAVA™~~Java~~ "to_string" method which is conventionally used. This means that the costly overhead associated with repeatedly invoking JAVA™~~Java~~ method "to_string" is avoided. In other words, operations that are conventionally performed each time the JAVA™~~Java~~ "to_string" method is invoked need not be performed. As a result, the performance of virtual machines, especially those operating with limited resources (e.g., embedded systems) can be improved.